Des modèles aux programmes à base de composants : Besoin de langages à composants

Luc Fabresse - Ecole des Mines de Douai

Les composants logiciels sont des entités logicielles réutilisables promettant une réduction des coûts liés au développement, à la maintenance et à l'évolution d'un logiciel. Actuellement, de nombreuses propositions se réclament du mode de développement par assemblage de composants logiciels. Malgré un vocabulaire parfois commun (composant, port, interface, service, connexion, connecteur), ces propositions sont disparates de par leurs origines, leurs objectifs, leurs concepts ou encore leurs mécanismes. De plus, elles restent difficiles à utiliser en pratique par manque de véritables langages de programmation sémantiquement fondés et concrètement utilisables. Devant tant de profusion, nous nous intéressons, aux problématiques suivantes: qu'est-ce qu'un langage à composants? Quels sont les avantages de ces langages? Comment réaliser un langage à composants? Dans ce contexte, nous proposons Scl, un langage de programmation à composants permettant de mettre en pratique réellement la programmation par composants (PPC).

De par sa conception, Scl se veut être un langage : (i) minimal dans le sens où toutes les abstractions et tous les mécanismes qui lui sont intégrés répondent à un besoin identifié ; (ii) simple car ses abstractions et ses mécanismes sont de haut niveau ; (iii) détaillé car nous avons abordé un ensemble de questions souvent oubliées dans les autres propositions comme l'auto-référence, le passage d'arguments ou le statut des composants de base (collections, entiers, etc) dans un monde unifié ; (iv) dédié à la PPC, car prenant en compte les deux préoccupations que nous jugeons majeures en PPC: le découplage et la non-anticipation.

Le coeur de Scl repose principalement sur les concepts de composant, de port, de service, de connecteur et de « code glue » ainsi que sur les mécanismes de liaison de ports et d'invocation de service.

Afin d'améliorer encore la séparation des préoccupations au niveau du code, Scl intègre une unification des concepts d'aspect (issu de la programmation par aspects) et de composant. et propose un ensemble de différents types de liaisons de ports qui permettent d'utiliser un même composant de façon standard ou de façon transversale.

Scl intègre aussi un mécanisme général permettant d'établir des connexions entre composants basées sur les changements d'états de leurs propriétés sans que leurs programmeurs n'aient à écrire une ligne de code spécifique à cet effet. Deux prototypes de Scl existent, le premier et le plus abouti est écrit en Smalltalk et le second en Ruby.